DEVELOPMENT OF A SMALL AND INEXPENSIVE TERRAIN AVOIDANCE SYSTEM FOR AN UNMANNED AERIAL VEHICLE VIA POTENTIAL FUNCTION GUIDANCE ALGORITHM

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Shane Alan Wallace

September 2010

COMMITTEE MEMBERSHIP

TITLE:	DEVELOPMENT OF A SMALL AND INEXPENSIVE TERRAIN AVOIDANCE SYSTEM FOR AN UNMANNED AERIAL VEHICLE VIA POTENTIAL FUNCTION GUIDANCE ALGORITHM
AUTHOR:	Shane Alan Wallace
DATE SUBMITTED:	September 2010
COMMITTEE CHAIR:	Dr. Robert A. McDonald
COMMITTEE MEMBER:	Dr. Jordi Puig-Suari
COMMITTEE MEMBER:	Dr. Eric Mehiel
COMMITTEE MEMBER:	Dr. Lynne Slivovsky

ABSTRACT

DEVELOPMENT OF A SMALL AND INEXPENSIVE TERRAIN AVOIDANCE SYSTEM FOR AN UNMANNED AERIAL VEHICLE VIA POTENTIAL FUNCTION GUIDANCE ALGORITHM

Shane Alan Wallace

Despite the first unmanned aerial vehicle (UAV) mission being flown on Aug 22 1849 to bomb Venice [(1)] UAVs have only recently began to modernize into sophisticated tools beyond simple aerial vehicles. With an increasing number of potential applications, such as cargo delivery, communications, search and rescue, law enforcement, and homeland security, the need for appropriate UAV technology advancement also arose. Here, the development of a low-cost collision avoidance system is described. Hardware was tested and selected based on predetermined constraints and goals. Additionally, a variety of potential functions were explored and assessed at their effectiveness in preventing a collision of a UAV with mountainous terrain. Simulations were conducted using Cloud Cap's Piccolo autopilot in conjunction with Matlab. Based on these simulations, a set of potential functions was selected to be used with the chosen hardware on subsequent UAV-development-related projects.

Keywords: Laser Scanner, Potential Function Guidance, Terrain Avoidance, and UAV.

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

APF	=	Artificial Potential Field
ARF	=	Almost Ready to Fly
AVL	=	Athena Vortex Lattice
CFIT	=	Controlled Flight Into Terrain
$\nabla\tau$	=	Gradient of the Potential Function
GPWS	=	Ground Proximity Warning System
ISR	=	Intelligence/Surveillance/Reconnaissance
τ	=	Potential Function
σ	=	Potential Function Constant
PWM	=	Pulse-Width Modulation
RADAR	=	Radio Detection & Ranging
RC	=	Radio Controlled
SLAM	=	Simultaneous Localization and Mapping
SONAR	=	Sound Navigation & Ranging
TFR	=	Terrain Following Radar
UART	=	Universal Asynchronous Receiver/Transmitter
UAV	=	Unmanned Aerial Vehicle

Chapter 1: Introduction

1.1 Benefits of Terrain Avoidance

Today, unmanned aerial vehicles (UAVs) are being more frequently used by the government, military and civilian groups. They can perform a variety of missions including Intelligence/Surveillance/Reconnaissance (ISR), cargo delivery, communications, search and rescue, electronic warfare, use as lethal munitions, crop dusting, environmental and forestry monitoring, law enforcement, homeland security, and aerial surveillance more inexpensively and with less risk than a manned aircraft. Although UAVs will be operating in diverse environments ranging from urban areas to mountainous landscapes, all environments present similar hazards to the UAV: unknown terrain and obstacles.

Imagine a small UAV flying at 1,000 feet or higher because of unknown terrain to get an image of a target vehicle. With a given camera, the quality of the image may only be detailed enough to recognize the type of vehicle and color. If a higher detailed image is desired, a higher resolution camera could be installed but would significantly increase the weight, volume requirements, and potentially the cost of the system. Aircraft vibrations could also hinder the performance of a higher resolution camera, necessitating the use of a sophisticated image stabilization system. The use of a terrain avoidance system, however, would allow the UAV to fly at one hundred feet, allowing for the capture of an image with a resolution ten times better than before. The original camera can now capture an image with enough detail to determine the number of occupants, the license plate number and other desired information.

The concept of terrain avoidance has been around long before there were UAVs. In the late 1970s, aircraft manufacturer Boeing developed the term “Controlled Flight

Into Terrain" (CFIT) which describes the act of flying a perfectly good aircraft into the ground [(2)]. To prevent CFIT, Aircraft manufacturers developed the Ground Proximity Warning System (GPWS) which is now FAA required equipment for all 10 or more passenger aircraft [(3)]. When the GPWS senses that the aircraft is too close to the ground, it gives the pilot an audio & visual warning message of the impending danger. This alerts the pilot to take the appropriate actions to avoid crashing [(4)]. UAVs would definitely benefit in having a terrain avoidance system like GPWS that also has the ability to command and fly the UAV around the obstacle autonomously.

Using a terrain avoidance system also has other benefits. Unlike a manned aircraft, UAVs have to be either flown by a highly trained pilot from a ground station with limited situational awareness or completely autonomously using only the data it was programmed with and information from on-board sensors. Installation of a terrain avoidance system will remove the need for the highly trained pilot and will therefore decrease operating cost. It will also increase reliability as humans tire and make mistakes. Instead, a less trained individual can input waypoints of a desired flight path into a ground station and the autopilot can be allowed to completely fly the aircraft. Since the operator will not need to worry about crashing into terrain, he or she will instead be able to focus his or her attention on the data being obtained as well as the overall mission of the UAV. The UAV will also not be limited to flying within radio range, instead it can fly in communication and even GPS denied environments making it much more versatile in the types of missions it can perform.

With the added ability of a terrain avoidance system, a UAV can also become a stealthier platform. By being able to fly low, it can use the mountains and buildings as

cover to hide its sound, radar and visual footprint. Central control and mission planning software that considers enemy detection while determining the best flight path, like AeroMech's SharkFin program [5], is already available and would benefit from having a terrain avoidance system on board the aircraft.

1.2 Overview of The Operation of an Terrain Avoidance Algorithm

A terrain avoidance algorithm starts by collecting terrain and obstacle locations through a number of possible sensors. Then, the locations are stored such that they require a small amount of memory and can be accessed quickly. The algorithm then uses the location of the aircraft and the obstacles to decide if it needs to take over control and deviate the plane's flight path. The process repeats itself over and over until the end of the mission as can be seen in Figure 1.

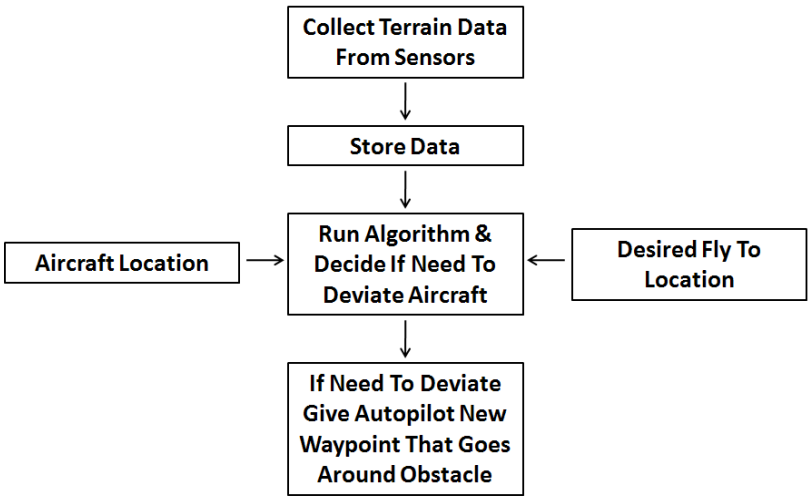


Figure 1 Operational Flow Of A Terrain Avoidance System

1.3 Objective of Thesis

Originally, the objective of this thesis was to develop and flight test a terrain avoidance system using a scanning laser ranger finder and a potential function algorithm that was small, inexpensive and easy to integrate onto a small UAV (10lb to 250lb

Range). After work began on the project, it was clear that the scope of the project was too large for one Master's thesis. The objective of this thesis was therefore narrowed to developing and running a Matlab simulation of the terrain avoidance algorithm and a method for storing the points. Also, a plan to obtain a flying testbed with a terrain avoidance system was devised.

On the following page Figure 2 shows the overall plan of the project that details the work I have done, the work I oversaw by Nick Utchig and Ryan Halper, as well as future work to be completed. The next chapter presents the hardware proposed and selected for the overall project as well as work that has already been completed by other students and myself. Chapter 3 describes in detail the method used to store the points found by the sensors. Chapter 4 covers the potential function algorithm and how the constant values were chosen. Chapter 5 provides the conclusion and future work that needs to be done to complete the overall goal of the project to have a flying aircraft that has a functional terrain avoidance system.

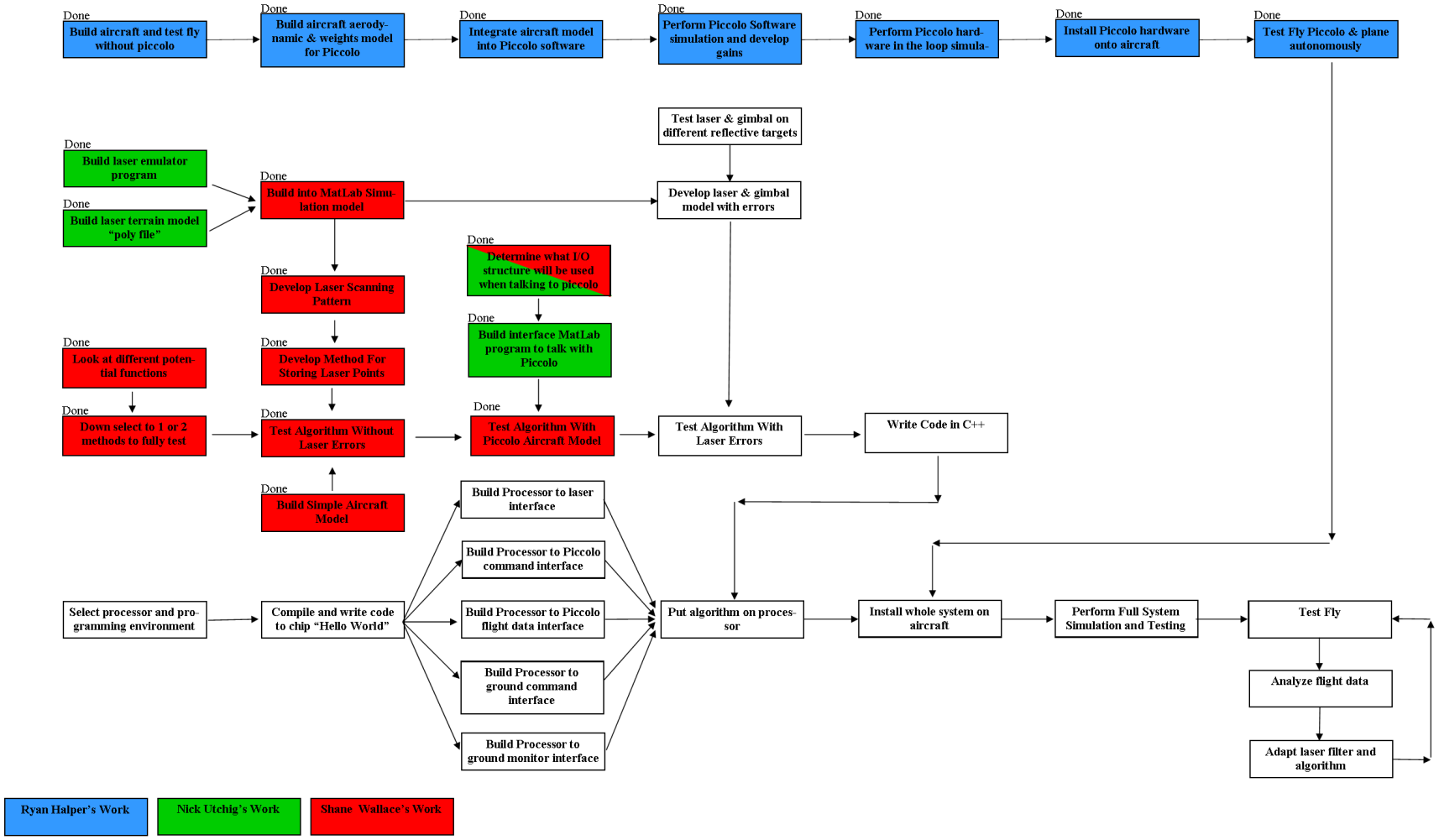


Figure 2 Overall Project Plan

Chapter 2: Hardware

2.1 Hardware Plan

Hardware was selected with the goal of building an inexpensive testbed for the algorithm and its components. After initial testing is completed and the terrain avoidance method is proven, more money and resources can be invested in lighter and higher quality components to develop a smaller, more compact system. The autopilot, aircraft testbed, laser and processor hardware components are discussed in the following chapter.

2.2 Autopilot

The following factors were considered during the selection of an autopilot:

- popularity of the autopilot
- ease of terrain avoidance system integration
- availability of support
- flight simulation software outputs
- setup time
- size
- weight
- cost

After examining many autopilot systems including Procerus's Kestrel Autopilot [6], MicroPilot's MP2028 Autopilot [7], and ONAVI's Phoenix Autopilot [8], Cloud Cap's Piccolo LT [9] was chosen because it aligned best with the goals of this project. First, it is currently in use by a number of different UAV manufacturers and users including Lockheed Martin, NASA, NAVAIR and AeroMech Engineering. Support is offered by a local aerospace company, AeroMech Engineering, who has helped with a number of Cal Poly projects and offered technical assistance with the set up and operation of the Piccolo. The system is a turnkey setup which is ideal because the focus of this project is on development of a terrain avoidance system and not on the autopilot itself. The overall autopilot system was relatively inexpensive and within the project budget. The included

Cloud Cap Ground Station software can simulate missions and produce position, orientation, velocity and mission data that can be sent to and analyzed by Matlab. The size of the autopilot is only 5.1 inches x 2.34 inches x 0.76 inches and its weight is 1.6 oz allowing for it to be easily integrated into a small aircraft. Figure 3, 4 and 5 shows the autopilot, ground station and ground station software.

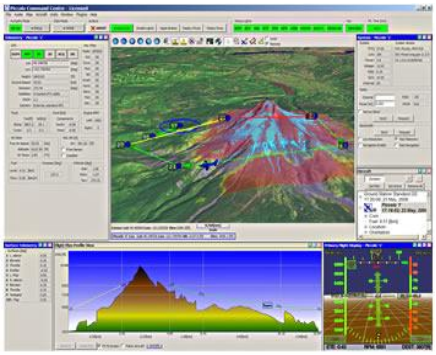


Figure 3 Piccolo LT [9] **Figure 4 Ground Station [9]** **Figure 5 Ground Station Software [9]**

The interface between the autopilot simulation software, "Piccolo Command Center," and Matlab was created by Computer Engineering student Nick Utschig. He developed a C program that takes the packets from the Piccolo Command Center simulator and parses out the desired information for use by the terrain avoidance algorithm. It then writes the data to a text file that can be easily read by Matlab using a file lock system to ensure data is not corrupted or lost. He also developed a method for integrating new waypoints into the autopilot using Matlab and text files. When a new course deviation is necessary, the Matlab script can communicate directly with the autopilot and command it around the obstacle.

2.3 Aircraft Testbed

A flight testbed is needed to test hardware components and the overall algorithm inexpensively and easily. Ryan Halper's senior project [10] involved building and autonomously flying the testbed using the Piccolo autopilot.

The aircraft used for the testbed, a Sig Kadet Senior ARF, was selected for the following reasons: cost, good flight characteristics, quick build time, availability of replacement parts, and payload flexibility. Because the Kadet Senior is a popular almost-ready-to-fly (ARF) radio controlled (RC) plane, 90% of the construction is already completed. The kit is a trainer aircraft, so it has docile flight characteristics making it an ideal test bed aircraft. Since it is a popular RC trainer, replacement parts will be easy to find should there be a mishap. It is not uncommon for the aircraft to have a 5lb or more payload, so it should meet the capacity requirements of any hardware that needs to be flight tested. To facilitate flight testing, an electric propulsion system was chosen that had plenty of power and long flight duration. A RimFire 50-55-650 motor, Phoenix 45 speed controller, and 3 cell 5500mAh lithium polymer battery were used in the aircraft.

For the first part of Mr. Halper's project, he assembled the test aircraft without the expensive flight hardware to prove that the test bed was airworthy and dependable. Below in Figure 6, the aircraft can be seen after the first flight test, before any flight test hardware was installed.

Figure 6 Assembled Sig Kadet Testbed

Next, Mr. Halper measured the aircraft components' sizes and weights in order to develop an aircraft aerodynamic model using Athena Vortex Lattice (AVL) [11] software developed by Mark Drela at MIT, as well as a moment of inertia model. The models were then uploded into the Piccolo Command Center Ground Station software to test the overall system and to develop the aircraft autopilot flight control gains. After testing had been completed, the Piccolo autopilot was installed in the Kadet and readied for flight tests. Figure 7 depicts the aircraft with all autopilot components installed, ready for the first flight test.

Figure 7 Sig Kadet With Integrated Piccolo

The aircraft was autonomously flown on July 1, 2009 three times at a flight weight of 10lb 3oz for a total autonomous flight time of 9.5 minutes performing holding patterns and waypoint navigation. Thus, a flight proven testbed exists to test hardware and an algorithm when the project reaches that step.

2.4 Sensor Selection

When it comes to sensor selection, UAVs are constrained by payload weight, payload power, computing power, electrical power and cost that can be quite demanding on the design and implementation of a terrain avoidance system. Small UAVs are generally electrically powered through batteries which contain a limited amount of electrical energy. The more power a particular sensor requires, the more it will reduce the overall endurance and capability of the UAV. Computing power is also a major limiting factor because there typically is very little excess capacity on the autopilot microcontroller and it is typically utilized by the payload. More computational capacity can be obtained by adding more microcontrollers or even a small computer like the popular PC-104, but this reduces payload capacity by volume and weight, adds cost, and requires more electrical energy. To accommodate for these constraints, sensors must be light and compact in size. The cost of the sensor is also a major factor because to be competitive in today's market, one must also have a cost competitive product.

Possible terrain detecting sensor solutions can be broken down into two groups: passive and active. Passive sensors include optical flow sensors, monocular or stereo vision systems, and infrared cameras [(12)]. Passive sensors utilize the sun's reflected energy off of the target for optical detection. Because of this, a passive sensor is impractical for a terrain avoidance system as it will nullify operation in the night time

environment. Active sensors, on the other hand, provide their own energy source to locate the target and the transmitted energy used to detect obstacles is invisible to the human eye. They include Radio Detection & Ranging (RADAR), Sound Navigation & Ranging (SONAR) and Laser Range Finder [12]. All operate on the time-of-flight principle by sending a pulse towards a target and measuring how long it takes the pulse to be reflected by the target and return to the detector.

RADAR uses electromagnetic waves to identify position and speed of moving and non-moving objects by measuring how long a transmitted radio wave takes to return after it bounces off the target. There are a number of manned aircraft which are terrain-following radar (TFR) including the F-16, B-52, and B-1B. Research is being conducted to implement a radar system small enough for small UAVs by the University of Sydney [12]. Currently, they are flying a radar unit that consumes 3.7 watts, weighs 0.67 lb and has a range of 30 to 50 feet [12] that can be seen in Figure 8 [12].

Figure 8 University of Sydney Radar Range Finder [12]

SONAR uses sound propagation to acoustically locate and measure the distance to a given target. It was initially developed for underwater use and has been primarily advanced for use by seagoing vessels to scan underwater. Sensors have also been

developed for use above the water and are becoming widely used by many small unmanned land vehicles. These sensors, as shown in Figure 9, have very little current draw, are fairly small in size, and have a maximum range around 25 feet.

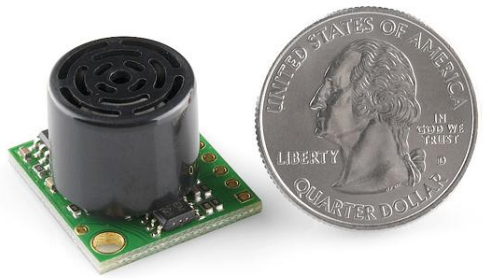


Figure 9 SONAR Range Finder [13]

Laser range finders have a beam that is usually eye safe and in the 600-1000nm wavelength range. By mounting the laser on a simple two axis gimbal, it is easy to construct a three dimensional map of the surrounding area. Laser range finders are being used more and more in autonomous operations. In the DARPA grand challenge, a number of vehicles used high end laser scanners to determine the location of the road. A number of setups used in the competition can be seen in Figure 10 [14]. Lasers range in size from 0.5 in^3 to 0.5 ft^3 and weigh anywhere from 0.5 to 29 lb depending on the accuracy and sample rate of the laser.

Figure 10 Laser Range Finders Used In DARPA Grand Challenge [14]

For this thesis, a laser range finder was selected to gather data for three reasons. First, a laser system is the most compact off-the-shelf device available for the distance we want to detect. They are also very affordable and have shown promise of performing as needed in past projects. Third, we were also able to obtain a 600 yard laser and gimbal to use easily and inexpensively. Although this thesis utilizes a laser to test the algorithm, the method for storing obstacles and avoidance algorithm will successfully operate with other sensor or sensor combinations with only slight modification.

2.5 Sensors

There are many different kinds of lasers that can be used to ascertain the distance to obstacles, ranging from the very expensive to the relatively inexpensive. Before one can be chosen for a given aircraft, the following factors need to be taken into consideration: scan rate, accuracy, range, cost, weight, size and power consumption. Since the goal is to develop a small and light weight terrain avoidance system, the size and weight are important factors. The project budget has limited us to spending a maximum of one thousand dollars for a laser and gimbal system. Because we are developing the algorithm and system, we do not know the exact required specifications such as accuracy and scan rate needed for the avoidance algorithm to be able to perform. A range of 200 yards or more at a scan rate of 100Hz or more were determined to be good starting points in requirements to find a laser. The Opti-Logic RS 800 [(15)] mounted on a Servo City SPT200 [(16)] gimbal was found to be a good starting point for the current known requirements. The Laser cost $695.00 which is under our budget limit and has a range of 800 yards with a scan rate of 200 Hz. The data is output on a standard RS-232 serial output. It weighs 8 oz., consumes less than 1.8 Watts and is only 1.3” x 3” x 3.3” in

size. The gimbal is controlled using RC servos threw a Pulse Width Modulation (PWM) signal. It has a 90 degree travel in pan and tilt and can be built up for under $100.00. Shown below in Figure 11 are the laser and gimbal components.

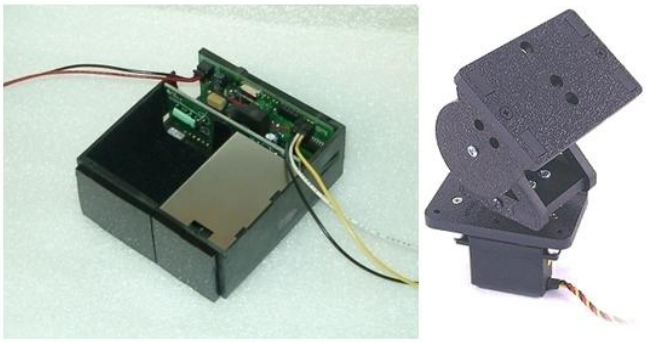


Figure 11 Laser [15] and Gimbal [16]

A popular choice for vehicles in the 2005 DARPA Grand Challenge was the Sick laser mounted on a tilt axis shown in Figure 12. It has a spinning mirror which allows a laser to scan in a single plan requiring the added tilt axis to be able to scan in 3D. This setup is 6" x 6" x 8", weighs 10 lb, consumes 20 watts, has a range of 80 yards and costs $5,800.00, making it bigger and heavier with higher power consumption than can be put on a small UAV. Because we are not trying to map the terrain as we go, and we do not have the option to slow down or stop when the terrain is bad, a laser of this size, scan rate, and accuracy is not appropriate for our situation.

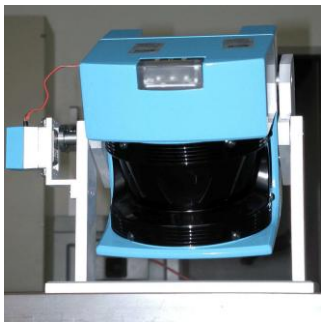


Figure 12 Sick Laser [17]

In the future when the exact specifications of the laser needed are known resources can then be put into developing a smaller, lighter and better performing laser and gimbal system. A possible future starting point is the Velodyne HDL-64E laser shown in Figure 13. It uses a 64-element sensor that spins at 300 -900 RPM. It is 10" x 8" x 8", weighs 29lb, and costs $79,000 [18]. Since we are not trying to map the terrain, it may be possible to decrease the number of sensors and instead have the unit fixed with the mounted sensors pointing in different directions. This would remove any mechanical scanning mechanism and hopefully produce a lighter, more robust and cheaper laser scanner.

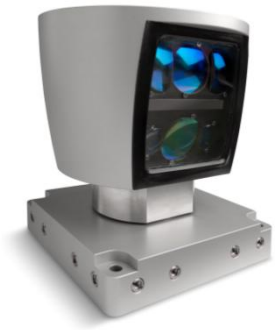


Figure 13 Velodyne Laser [18]

To be able to run a simulation of the algorithm, a laser simulator must be developed that has the capability to operate while leaving enough processing power for the computer to actually run the algorithm and any other needed code and software. The speed of the simulation needs be able to run at the same rate of the laser (200 Hz). This turned out to be more difficult than it was initially thought it would be.

An early laser simulation was developed using Matlab that was based on equations derived using basic trigonometry. It would look at every panel in a made-up terrain map and determine if a laser fired in a certain direction would hit the panel. Then,

it would calculate the distances to each of the panels it did hit and determine which was the closest and then report back the distance. Unfortunately it ran too slow for our application. An image of an early laser simulator in operation can be seen in Figure 14.

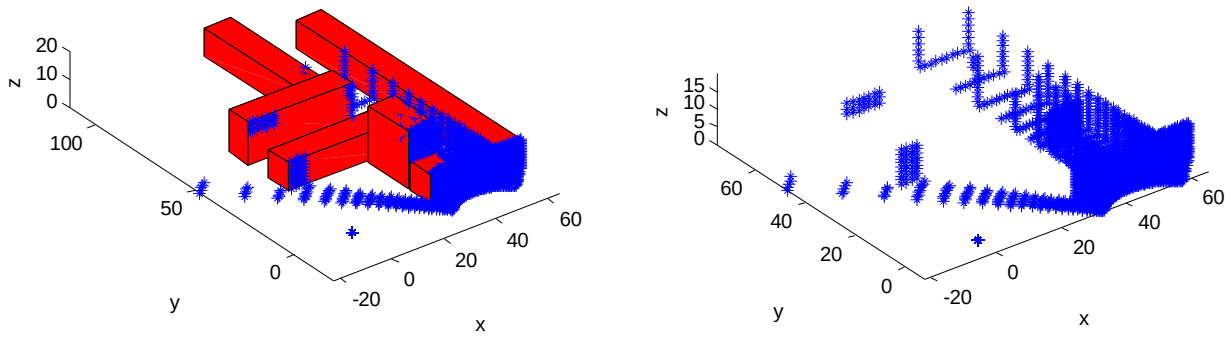


Figure 14 Early Laser Simulator

Nick Utchig, a computer engineering student, developed a laser simulation code that was able to run real time. Figure 15 shows the speed at which the code can look at a given number of triangles, determine which triangle the laser is pointing at, and determine the distance between the triangle and UAV.

Number of Triangles	Rate
1	9752 Hz
10	5162 Hz
100	1055 Hz
400	208 Hz
500	149 Hz
1000	47 Hz
5000	2.58 Hz

Figure 15 Run Times

The code was written in C and could be called by the Matlab simulation code. The algorithm used was based off of a ray tracing algorithm [19] that is typically used for computer graphic applications. Data was taken from the "Lat/Lon to Elevation" [20] web

page and used it to construct maps of areas around San Luis Obispo for simulations as shown in Figure 16.

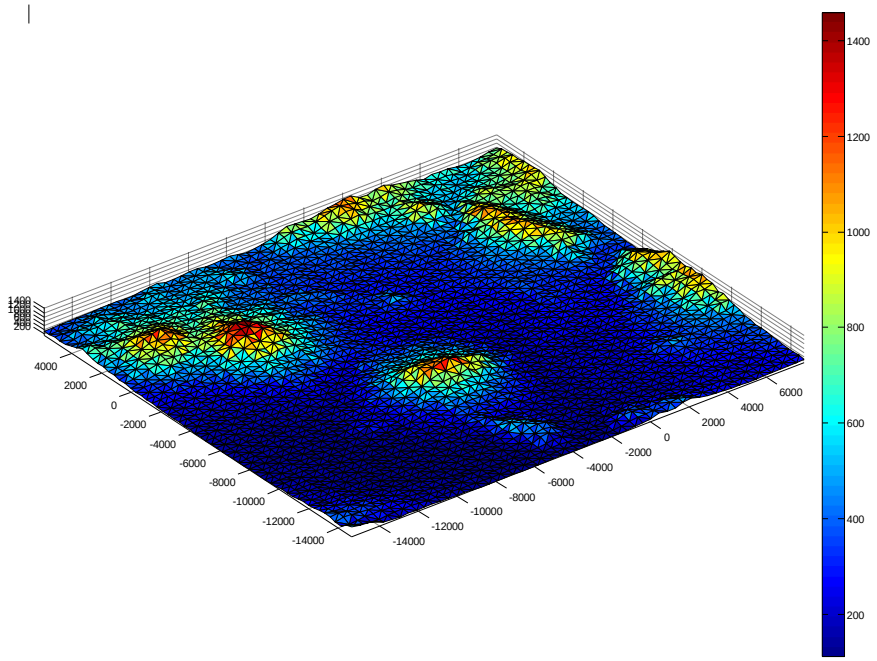


Figure 16 SLO Map Used To Generate Laser Distances

Currently the laser simulator does not contain the angle, position and distance error that the real laser and gimbal will have. When testing is moved to hardware in the loop testing the amount of error and its effect on the algorithm will have to be understood and accounted for.

Testing of the physical laser system started, but did not progress too far because more effort was focused on testing the algorithm and working on the simulation. The laser and gimbal were mounted on a cardboard box and connected to an Atmega 128 processor as shown in Figure 17. C code written onto the processor made the gimbal do a simple sweep in the pan and tilt. Then, the distance was read in and sent via serial to the computer where Matlab analyzed the data. A scan of the inside of a room was done and can be seen in Figure 18. This testing showed that the commanded angle was not

precisely the angle at which the reading was taken. For this reason, if readings are going to be taken while the gimbal is moving, the flight hardware is going to have to have the potentiometers in the servo calibrated so that the direction the laser reading is known.

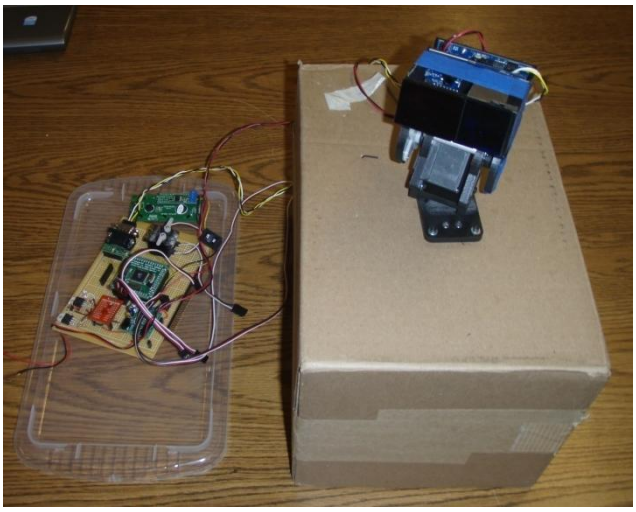


Figure 17 Laser & Gimbal Test Setup

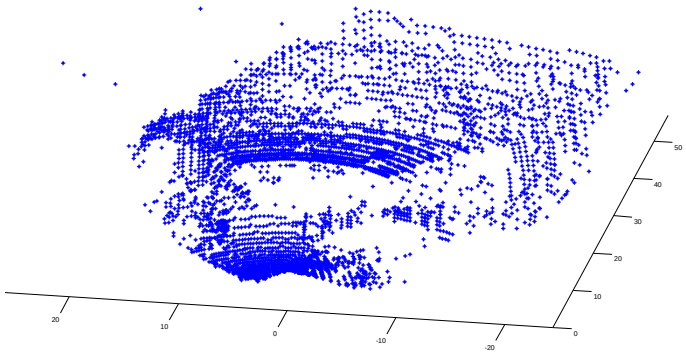


Figure 18 Scan of the Inside Of The Room

2.6 Processor

When work first began on the project, the idea was to develop the code on the processor. That way, it would be ready for flight testing when it was completed and simulations could be run simply by sending it data that it would normally receive from

the various sensors. An Atmega 128 was selected and efforts were spent to begin making various functions and libraries that would be needed to set up the processor and compile the required information for the algorithm. After a substantial amount of work, it was determined that the Atmega 128 was not going to work because it was too slow and did not contain enough memory to store the grid of points. It also took considerable time to develop code because it had to be compiled and then uploaded to the chip. Also when it ran, there was no easy way to visualize errors and their sources.

A meeting was set up with Nick Brake, a controls engineer at AeroMech Engineering [(21)], to determine what they normally do when developing programs and hardware for UAVs and what direction should be taken with the project. He suggested that the best course of action was to code and test the algorithm in Matlab to allow for the easiest and quickest development. Then after the code was tested and requirements became known for the hardware, a processor can be selected. He suggested looking at the Blackfin processor family, that can be seen in Figure 19 when the time comes. The code already written in Matlab can be compiled using Matlab Real Time Workshop and uploaded onto the processor. Technical help will also be available through AeroMech as they use the chip family on a number of their products.

Figure 19 Black Fin Processor [(22)]

2.7 Summary

In summary the hardware recommended and used for the project is a Piccolo Lt Autopilot and ground station, an Opti-Logic RS 800 laser and Servo City SPT200 gimbal, Sig Kadet aircraft test bed and a Black Fin processor. The final hardware used on a vehicle will in the end depend greatly on the vehicle itself, how low it will be allowed to fly to the ground and the amount of money willing to be spent to save weight and power.

Chapter 3: Storing of Laser Points

3.1 Storing of Laser Points

Point storage and calling greatly affects the speed, memory requirements, and success of an avoidance algorithm. If the algorithm takes too long to run, it will detect a needed course deviation after it's too late, leading to the vehicle crashing. Memory and processor requirements are of utmost importance when selecting and developing a method because of the UAV's limited computational resources. Three methods were found and investigated for solving the problem of storing points: an array, octree, and 2D grid structure.

The simplest method, an array structure, was the first method considered. All points found by the laser are simply put into a forever growing list. Two major problems quickly surfaced. First, there is an ever growing memory requirement, unless a search is performed to throw out old and unneeded points. Second there is no method to find points in a given location without searching through the whole list. This method was quickly abandoned.

The next method considered was an octree structure, which stores points in a recursively subdivided space. The method works by taking the volume of points and dividing it into eight cube-shaped volumes that are then each divided into another eight cube shaped volumes. This continues until the number of points contained in the cube is below a set threshold value [23]. An example of this method can be seen in Figure 20. The main benefit is that it allows for quick lookup of data in a given area without having to search through all the points. Like the array structure method, using an octree structure also has the problem of a forever growing memory requirement unless old points are searched for and removed. Since it is unknown where the points will be clustered, and

new points will be constantly added, cubes will have to be continuously divided. This method is therefore to difficult, complex and expensive to implement.

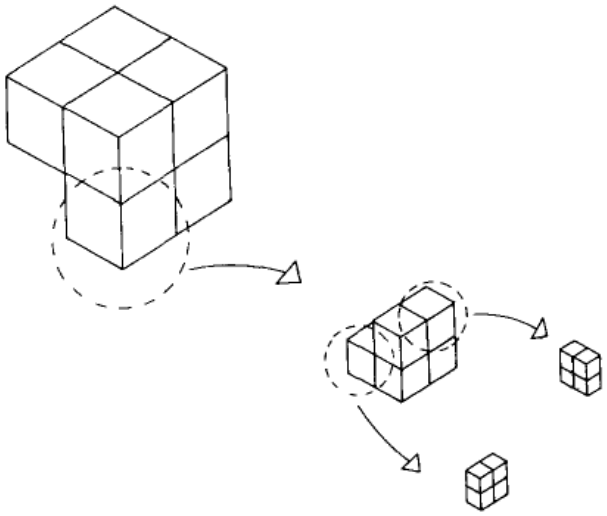
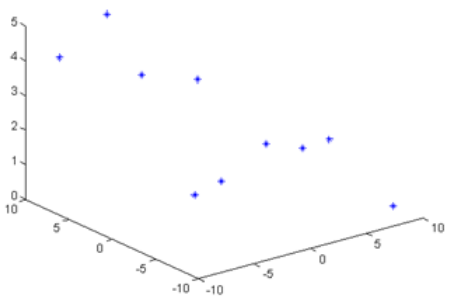


Figure 20 OCTREE Method [23]

The last method found incorporates the use of a 2D grid structure that stores the 3D data by taking the maximum height values of a given area and storing it in a node of the matrix. Containing the data within a matrix structure allows for very quick searching and placement of points. When a given row or column is a given distance away from the vehicle, it is simply deleted and no longer used. This allows the matrix to be a pre-allocated user defined size leaving the memory requirements not only known but also controllable and constant. This makes this method very desirable for use when developing a terrain avoidance algorithm to be installed on a small UAV and therefore was chosen to move forward with. An example of this method is shown in Figure 21.

x	y	Z
-3.1	5.6	3.5
1.8	-2.5	2
-1.4	-1.5	1.1
-0.1	-9.2	2.8
-9	7.4	4.3
-3.2	9.5	4.8
9.8	-6.8	0
-5	-3.1	1.2
5.1	-5.5	2.2
-4.8	-3.1	4.5



	16	13							
	15	9					15	15	
	12				27		14	12	
				16	35				
				25	48	35			
				15	15				

Figure 21 2D Matrix Storing 3D Points Example

3.2 2D Method for Storing Points

The 2D matrix method implements a matrix which is fixed in the X and Y direction, with positive Y always pointing North and X pointing East. The aircraft is placed at the center of the matrix such that the aircraft can travel in any direction and obstacles found by the laser can be recorded and ready to use as shown in Figure 22. As the aircraft travels, the obstacles that are on the grid are shifted the distance and direction the aircraft travels so that the aircraft is always in the center of the grid.

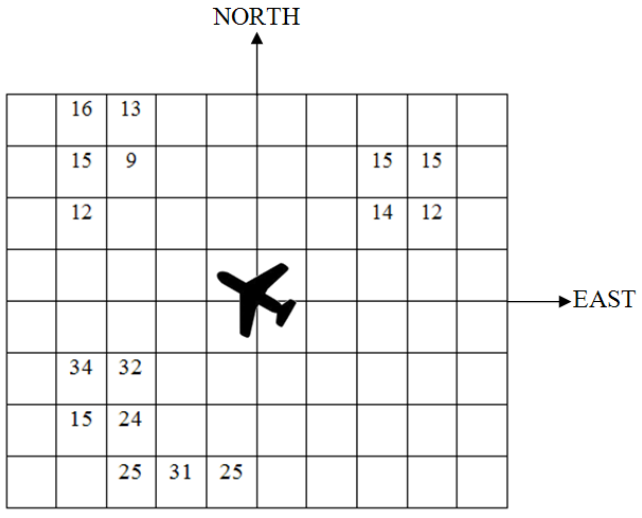


Figure 22 Matrix

3.3 Making the Matrix

When making the matrix to store the points, three factors need to be considered: the amount of memory required, grid resolution and ground dimensions that the matrix covers.

First, the memory required must be calculated so that it can be determined if it is capable of running on the memory available on the size of processor desired. To determine the memory requirements, Equation 1 was derived by first taking the grid size and dividing by the grid resolution to get the number of points along one edge. Then, the resulting number was squared to calculate the total number of points in the matrix. Because each point will be given a short data type on the processor, it will take up two bytes. Therefore, the number of points in the matrix was multiplied by two, producing the total bytes required to store the matrix.

$$bytes = (Grid\ \text{Length}/Grid\ Resolution)^2 * 2 \qquad \mathbf{1}$$

To determine how much ground distance is needed between each node, one must look at how close the UAV will be required to fly to objects. For testing of the algorithm, a grid that would allow the UAV to fly through city streets was used. According to the Federal Fire code, the minimum street width is 20 feet [(24)]. By putting four points between the street edges as seen in Figure 23, the spacing distance between each node is determined to be 5 feet. The number of points needed between two obstacles thus far has been an educated guess; future work will need to be done to determine if more or fewer points are appropriate depending on the UAV flight characteristics.

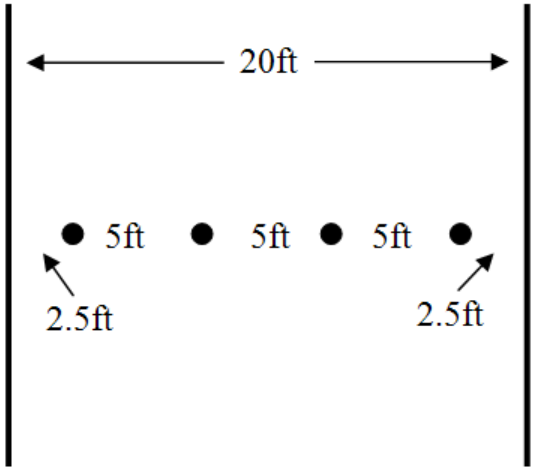


Figure 23 Grid Width Setting

The distance the matrix covers can be determined by examining how far away the laser can detect obstacles and how far the potential function needs to be able to look ahead. For the Opti-Logic RS800 laser, the ideal distance it can observe a target from is 2400 feet, which would make the matrix 4,800 by 4,800 feet in size. This will make an unrealistic matrix size to store for two reasons. First, the 2400 feet is the ideal distance of the laser for a perfect reflective target. Since we are primarily going to be calculating distances of non-reflective targets like vegetation and buildings, the performance will be nowhere near the ideal value. The second reason is that the calculated memory required of the grid with a five foot resolution would be 1,843,200 bytes, calculated from Equation 1. Cutting the distance down to 1000 feet should give the potential function enough distance to look ahead and turn the aircraft with an estimated 250 ft turn radius from simulations. This requires a matrix of 2,000 by 2,000 feet and only 320,000 bytes of memory, which is a more reasonable amount.

3.4 Storing the Points

Now that a grid has been created we are able to store the 3D points found by the laser into the 2D grid, but first the data from the laser has to be transformed into the same coordinate system as the grid. The laser returns a distance that is first converted to the aircraft coordinate system using the pan and tilt angles of the gimbal. Then, using the aircraft's heading, pitch, and roll angle it is converted to the grid coordinate system where +Y is fixed north. The nearest node on the grid is then found and only the highest point for that node is stored.

During testing, it was quickly discovered that storing only the highest point would create a problem. By only storing the highest points, vertical walls were not being modeled and the potential function would try and fly under the highest points thinking it was an acceptable path. A simple solution was created by storing the maximum and minimum points. This allows the potential function to make a wall of potentials that it will then not try and fly under. One downside to this fix is that it doubled the memory requirement to 640,000 bytes to allow for the storage of the two required matrices.

A limitation found in this method is that the UAV will never be allowed to fly under any obstacle because it will make a wall of potentials between the ground and object. One potential solution would be to store a three dimensional matrix of the laser points, but the memory requirements increase from 640,000 bytes to 128,000,000 bytes, requiring too much memory for a small processor.

3.5 Moving Grid

Now that the grid contains points, the fact that the aircraft is moving must be accommodated for by shifting the points in the grid. First, aircraft movement is tracked

by determining how far is has moved in the X, Y and Z direction since the last iteration. The resulting values are added to their respective variables ΔX, ΔY and ΔZ. Once ΔX or ΔY reaches the same distance or larger than the node spacing, all the points on the grid are shifted one unit in that direction and ΔX or ΔY is then subtracted by the node spacing distance. For the Z direction, the ΔZ values are applied every iteration. Below in Figure 24, a small grid can be seen with the grid shifting to the left one grid spacing because the plane traveled one node spacing distance to the right.

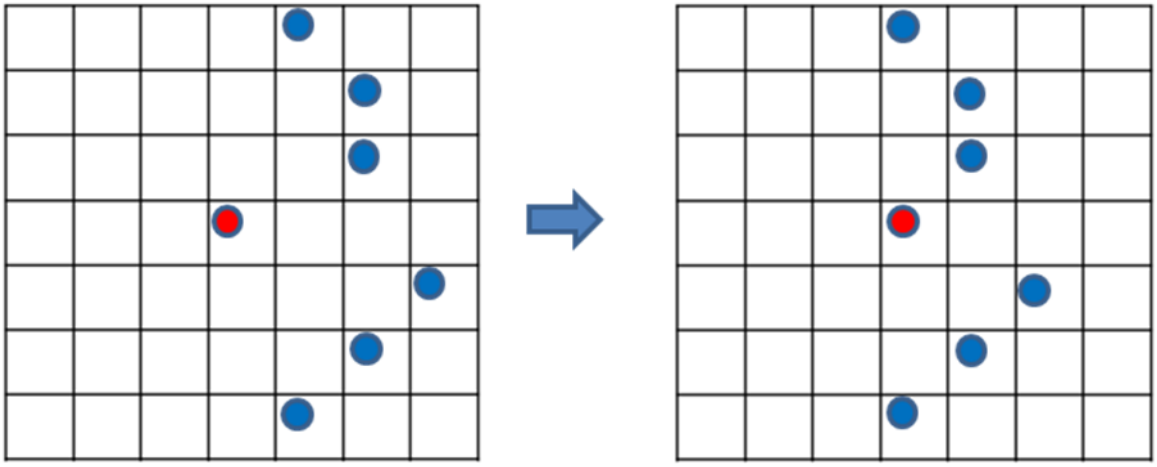


Figure 24 Grid Shifting One Space To The Left

Because this algorithm will ultimately be running on a microcontroller, a technique for shifting the grid should be developed so the stored data in memory does not have to actually be moved every time the grid is shifted. One possible solution is to instead move the memory address that points to the center of the grid

3.6 Storing Points Conclusion

Overall testing showed that using the 2D matrix method for storing the laser points works quite well and is quick and easy to implement. It has controllable memory requirement that is set by the grid resolution and size. One down side of the method is the

inability of the UAV to fly under an obstacle like a bridge or through a tunnel; a way to modify the method would be to use a 3D matrix that can model the area in 3D. There are numerous methods that could have been used, but this one is ideal for our set of requirements.

Chapter 4: Potential Function

4.1 Terrain Avoidance Algorithm

When developing a terrain avoidance algorithm, three factors are considered: amount of computational power needed, speed of algorithm and how easily it can be integrated into an autopilot system. As discussed before, the amount of processing power available is very limited because the algorithm must run on the same microcontroller as the autopilot and payload. The speed at which the algorithm can run is also important; if it takes too long to determine that a course deviation is needed, the UAV could crash before it deviates around the obstacle. The algorithm must also be easy to integrate such that the terrain avoidance system can be an add-on to an already functioning UAV system.

There are many collision avoidance methods that can be used to devise a terrain avoidance system including: occupancy grid [25], receding horizon [26], optical flow [27], Simultaneous Localization and Mapping (SLAM) [28] and potential function [27]. Receding horizon and optical flow methods are used with vision based sensors and not the laser rang finder used on this project, so both methods were discarded. The occupancy grid and SLAM method are used to not only avoid the obstacles but to also map the terrain in great detail. Because mapping terrain is not the goal of this project, and we do not want to pay the processing and memory penalty to do so, the method was abandoned. Therefore the potential function algorithm was chosen for use in this thesis. It is thought that the potential function was the best choice because of its small processor requirements. Work on potential functions has also already be done at Cal Poly by Professor Dr. Eric Mehiel and graduate student Masamitsu Tsuruta in the thesis “An Integrated Formation Flight Algorithm via Potential Function Guidance and Biomimetics” [29] and Dr. Mehiel

continues to work with potential function guidance at Cal Poly. Using Potential Fields was first developed by Khatib for manipulation of robotic arms [30] and was later adapted for mobile robot platforms by J.C. Latombe [31]. This method comes from the idea of electric potential fields in physics, where the repellant charge forces are the obstacles and the attractive charge force is the point the vehicle is trying to navigate. A two dimensional representation of a field can be seen in Figure 25; the repellent force in red, the attractive force in green and the flight path in black. The arrows point in the direction of the summed potential gradients and the length of each arrow is the relative strength at that location. As you can see, if an object is at any point on the map it can travel in the direction of the arrows and will be lead to the goal without hitting the obstacle. Obstacle avoidance is achieved in this case by calculating the strength of the gradient potential field and determining if it is higher than a threshold value. If it is, the gradient of the field is taken, giving the best path to avoid the obstacle and still progress towards the given target. This is able to be accomplished because the effect of each obstacle potential gradient field is able to be added to the attractive field using superposition.

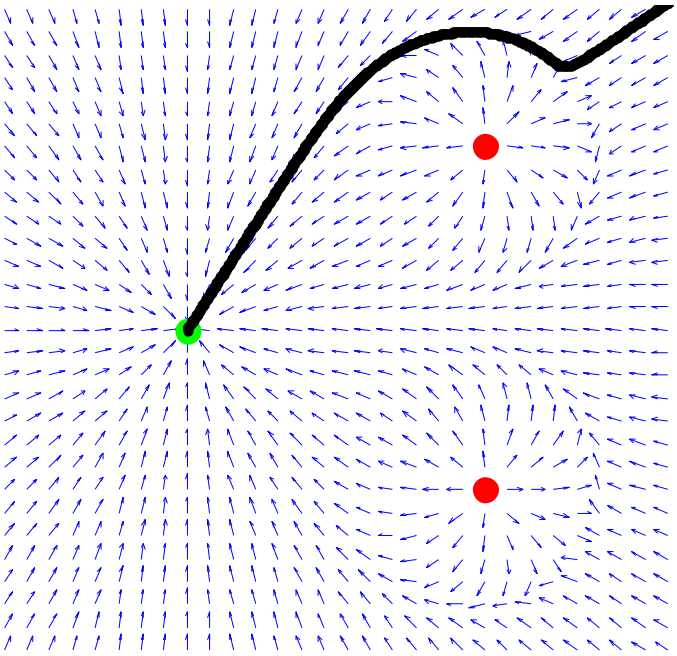


Figure 25 Potential Field Guidance Example

4.2 Potential Functions

Potential functions themselves are just simply scalar equations that are differentiable, but some equations are more effective than others. The equations used cannot be computationally expensive since it must be calculated every time the algorithm runs for every point being tracked. The decay rate, strength and shape of the potential equation to be used also has to be considered for the type of object being represented. For this thesis, seven potential equations τ and their gradients $\nabla\tau$ were closely investigated and are shown below in Equations 2 to 15. Plots of the potential strength and the magnitude of the gradient as a function of radius with the constants σ_i set to 1 are shown in Figure 26 to Figure 32.

Exponential Potential:

$$\tau = \sigma_1 * e^{\left[\frac{1}{\sigma_2}R^2\right]} \quad \textbf{2}$$

$$\nabla\tau = \frac{2\sigma_1}{\sigma_2} * e^{\left[\frac{1}{\sigma_2}R^2\right]}X\hat{i} + \frac{2\sigma_1}{\sigma_2} * e^{\left[\frac{1}{\sigma_2}R^2\right]}Y\hat{j}\frac{2\sigma_1}{\sigma_2} * e^{\left[\frac{1}{\sigma_2}R^2\right]}Z\hat{k} \quad \textbf{3}$$

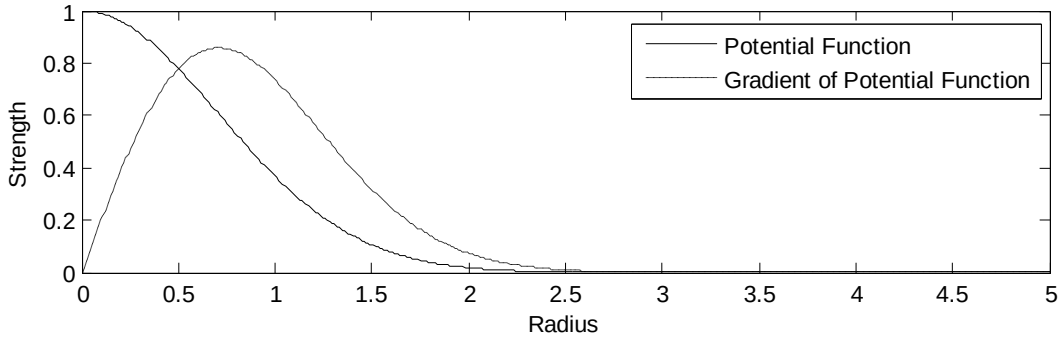


Figure 26 Exponential Potential Plot

Quadratic Potential:

$$\tau = \sigma_1 R^2 \quad \textbf{4}$$

$$\nabla\tau = 2\sigma_1 X\hat{i} + 2\sigma_1 Y\hat{j} + 2\sigma_1 Z\hat{k} \quad \textbf{5}$$

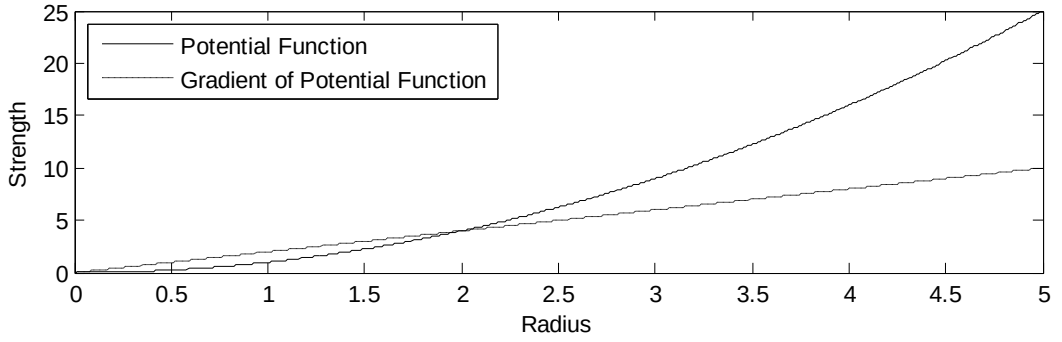


Figure 27 Quadratic Potential Plot

1/R Potential:

$$\tau = \sigma_1 * 1/R \quad \textbf{6}$$

$$\nabla\tau = -\frac{\sigma_1}{R^2} * X\hat{i} + -\frac{\sigma_1}{R^2} * Y\hat{j} + -\frac{\sigma_1}{R^2} * Z\hat{k} \quad \textbf{7}$$

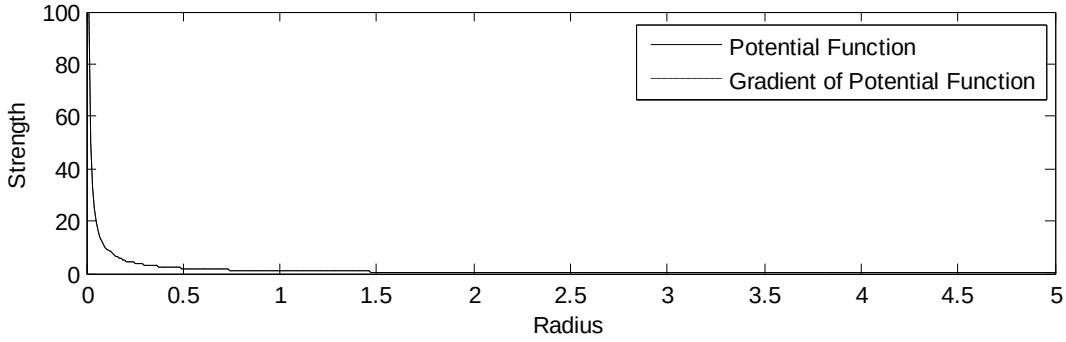


Figure 28 1/R Potential Plot

R Potential:

$$\tau = \sigma_1 * R \quad \textbf{8}$$

$$\nabla\tau = \frac{\sigma_1}{R} * X\hat{i} + \frac{\sigma_1}{R} * Y\hat{j} + \frac{\sigma_1}{R} * Z\hat{k} \quad \textbf{9}$$

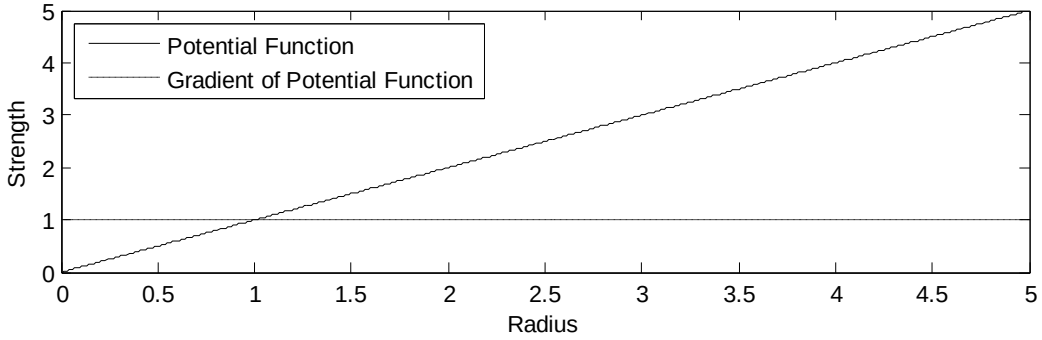


Figure 29 R Potential Plot

R^2 Potential:

$$\tau = \sigma_1 * R^2 \quad \textbf{10}$$

$$\nabla\tau = 2\sigma_1 X\hat{i} + 2\sigma_1 Y\hat{j} + 2\sigma_1 Z\hat{k} \quad \textbf{11}$$

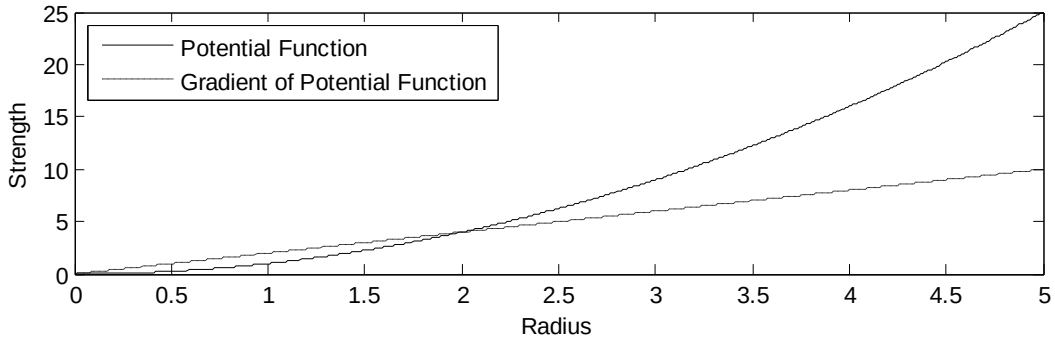


Figure 30 R^2 Potential Plot

R^3 Potential:

$$\tau = \sigma_1 * R^3 \quad \textbf{12}$$

$$\nabla\tau = 3\sigma_1 RX\hat{i} + 3\sigma_1 RY\hat{j} + 3\sigma_1 RZ\hat{k} \quad \textbf{13}$$

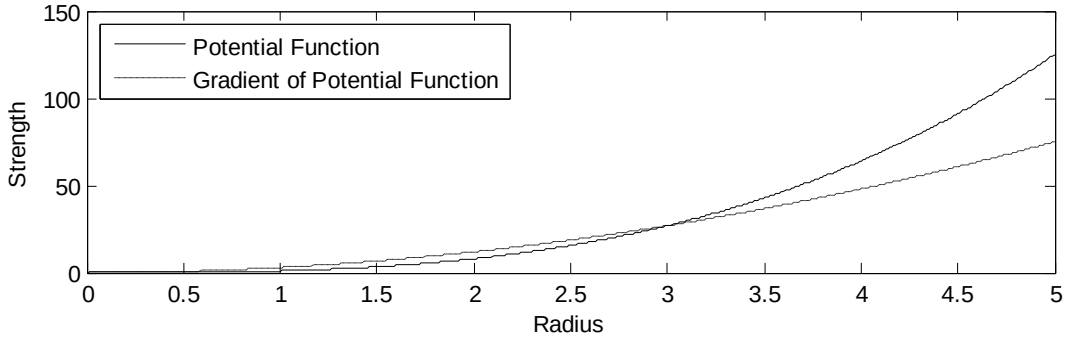


Figure 31 R^3 Potential Plot

R^4 Potential:

$$\tau = \sigma_1 * R^4 \quad \textbf{14}$$

$$\nabla\tau = 4\sigma_1 R^2 X\hat{i} + 4\sigma_1 R^2 Y\hat{j} + 4\sigma_1 R^2 Z\hat{k} \quad \textbf{15}$$

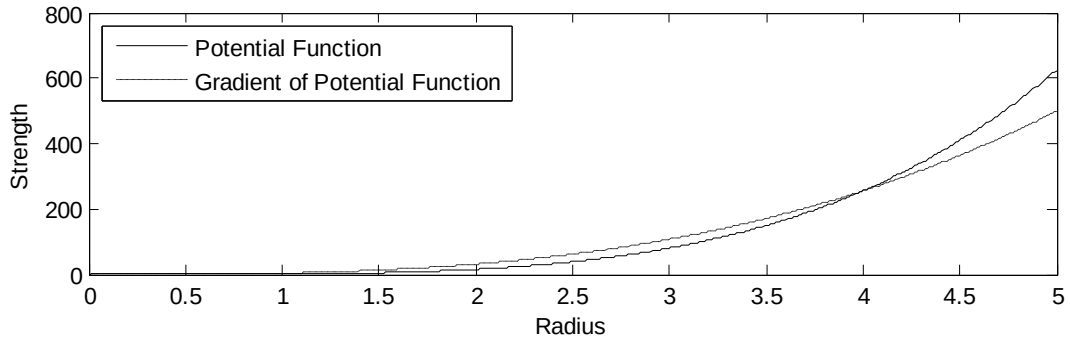


Figure 32 R^4 Potential Plot

For the goal location, a potential function gradient equation that is independent of distance is desired such that the aircraft has a consistent attraction force acting on it. From the different potential equations that were studied, the gradient of R shown in Equation 9 does exactly that. It can also be easily computed and gives a constant strength independent of location. A plot of the field it creates as can be seen in Figure 33.

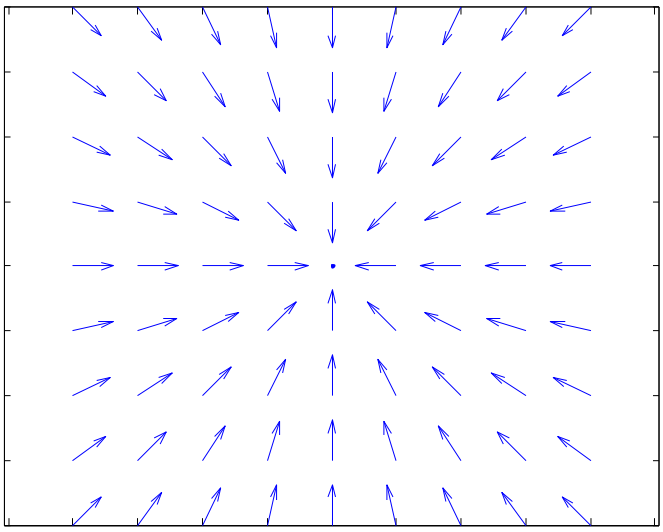


Figure 33 R Gradient Attraction Field Plot

Unlike the goal potential, the repulsive potential needs to have a strength that changes with distance from the obstacle. When near the obstacle, the repulsive force needs to be at its maximum. As you move away the strength drops until it reaches zero. Looking at the list of considered potential functions, the exponential potential function best represents the obstacle.

Typically, the gradient is used but if you look at the gradient of the exponential, you can see that it starts at 0 strength and increases until it reaches a strength of 0.85 at a radius of 0.7 and then declines. Since we want something more like the exponential function, it was found that if τ of the Exponential Potential Equation (Equation 2) was multiplied by $\nabla\tau$ of the R Potential Equation (Equation 9), a more desirable gradient potential was obtained as shown in Equation 16 and Figure 34.

$$\nabla\tau = \sigma_1 * e^{\left[\frac{1}{\sigma_2}(X^2+Y^2+Z^2)\right]} * \frac{\sigma_3}{R} * X\hat{i} + \sigma_1 * e^{\left[\frac{1}{\sigma_2}(X^2+Y^2+Z^2)\right]} * \frac{\sigma_3}{R} * Y\hat{j} + \sigma_1 * e^{\left[\frac{1}{\sigma_2}(X^2+Y^2+Z^2)\right]} * \frac{\sigma_3}{R} * Z\hat{k} \quad \textbf{16}$$

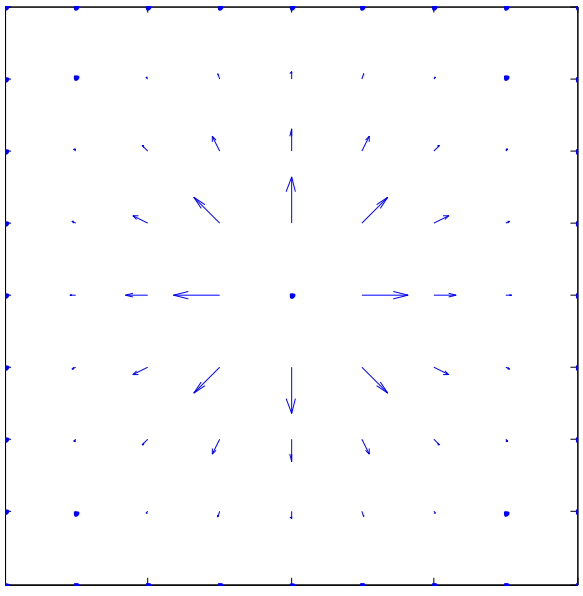


Figure 34 Exponential Repulsion Field Plot

During the development and testing of the algorithm, it was found that it is very desirable to have the decay rate of the repulsive potential different between the Z axis and the X and Y axis. The constants in the exponential function need to be set so that the repulsion force has a large enough effect that the aircraft has time to reroute in the X & Y direction and avoid hitting any obstacles. Unfortunately, if thee constants are the same in the Z direction, the aircraft will fly higher than is desired. Fortunately a simple method was developed to allow the decay rates to be different. By simply multiplying the Z value by a constant σ_4 that is greater than one, the calculated Z gradient is greater than the actual distance. This is shown below in Equation 17, 18 and in Figure 35 where σ_1, σ_2, σ_3 =1 and σ_4 =2.

$$R = \sqrt{X^2 + Y^2 + (\sigma_4 * Z)^2} \tag{17}$$

$$\nabla\tau = \sigma_1 * e^{\left[\frac{1}{\sigma_2}(X^2+Y^2+(\sigma_4*Z)^2)\right]} * \frac{\sigma_3}{R} * X\hat{i} + \sigma_1 * e^{\left[\frac{1}{\sigma_2}(X^2+Y^2+(\sigma_4*Z)^2)\right]} * \frac{\sigma_3}{R} * Y\hat{j} + \sigma_1 * e^{\left[\frac{1}{\sigma_2}(X^2+Y^2+(\sigma_4*Z)^2)\right]} * \frac{\sigma_3}{R} * \sigma_4 * Z\hat{k} \tag{18}$$

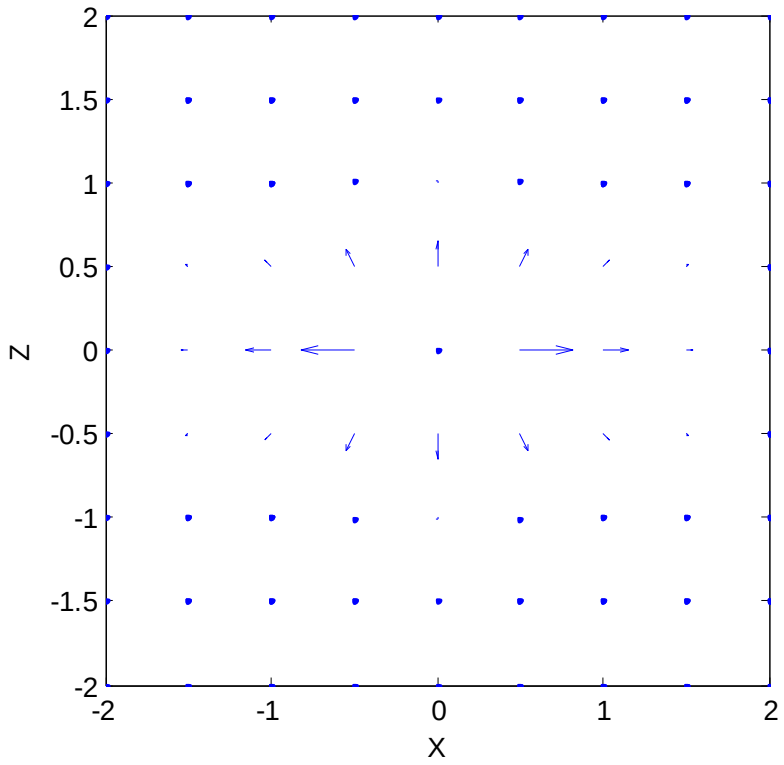


Figure 35 Exponential Repulsive Field Plot With Z Scaling

By taking the summation of all potential function gradients of the goal location and obstacle points, the best direction for the aircraft to travel can be determined. Equation 19 shows the equation used to do this. $\nabla\tau_g$ represents the goal potential gradient while $\nabla\tau_{oi}$ represents each obstacle potential gradient.

$$\nabla\tau = \nabla\tau_g + \sum_{i=1}^{n} \nabla\tau_{oi} \qquad \textbf{19}$$

4.3 Tuning The Potential Functions

Now that we have the two potential equations, constants must be selected to appropriately represent the goal or obstacle as needed. Also, the minimum repulsive value for the terrain avoidance algorithm to take control of the aircraft also needs to be

set. The following method was developed to choose conservative constants through trial and error.

We will start with σ_1 of the goal potential gradient (Equation 9) and use it as the basis for setting all other values. σ_1 of the equation could be set to any value since For simplicity, we will set it to one, making the gradient attraction field a unit vector. If you wanted to run this on a small vehicle size processor, you could increase the value to 1000 to avoid the use of floating point variables. Equation 20 is the final equation used for the attraction potential.

$$\nabla\tau = \frac{1}{R} * X\hat{i} + \frac{1}{R} * Y\hat{j} + \frac{1}{R} * Z\hat{k} \qquad \textbf{20}$$

The threshold value of the repulsive potential and terrain avoidance take-over also needs to be set. It was found that taking half the value of the attraction potential worked well and allowed for the repulsive potential to be easily set. For this setup, the threshold is 0.5.

For the repulsive equation, there are four constants that need to be set. We will start with σ_3, which is from the R gradient equation. Because σ_3 simply controls the overall strength like σ_1 from the exponential equation, we can simply set it to one and use σ_1 to set the strength.

$$R = \sqrt{X^2 + Y^2 + (\sigma_4 * Z)^2} \qquad \textbf{21}$$

$$\nabla\tau = \sigma_1 * e^{\left[\frac{1}{\sigma_2}\left(X^2+Y^2+(\sigma_4*Z)^2\right)\right]} * \frac{1}{R} * X\hat{i} + \sigma_1 * e^{\left[\frac{1}{\sigma_2}\left(X^2+Y^2+(\sigma_4*Z)^2\right)\right]} * \frac{1}{R} * Y\hat{j} + \sigma_1 * e^{\left[\frac{1}{\sigma_2}\left(X^2+Y^2+(\sigma_4*Z)^2\right)\right]} * \frac{1}{R} * \sigma_4 * Z\hat{k} \qquad \textbf{22}$$

Next we will look at σ_1, σ_2 and σ_4 for the repulsive exponential. As stated before, σ_1 controlls the overall strength of the potential, while σ_2 controls the distance that it has an impact on and σ_4 controls the scaling of the exponential in the Z direction. Figure 36 is a plot of the exponential magnitude as a function of radius in the X and Y directions to illustrate the effects of changing $\sigma_1$and σ_2.

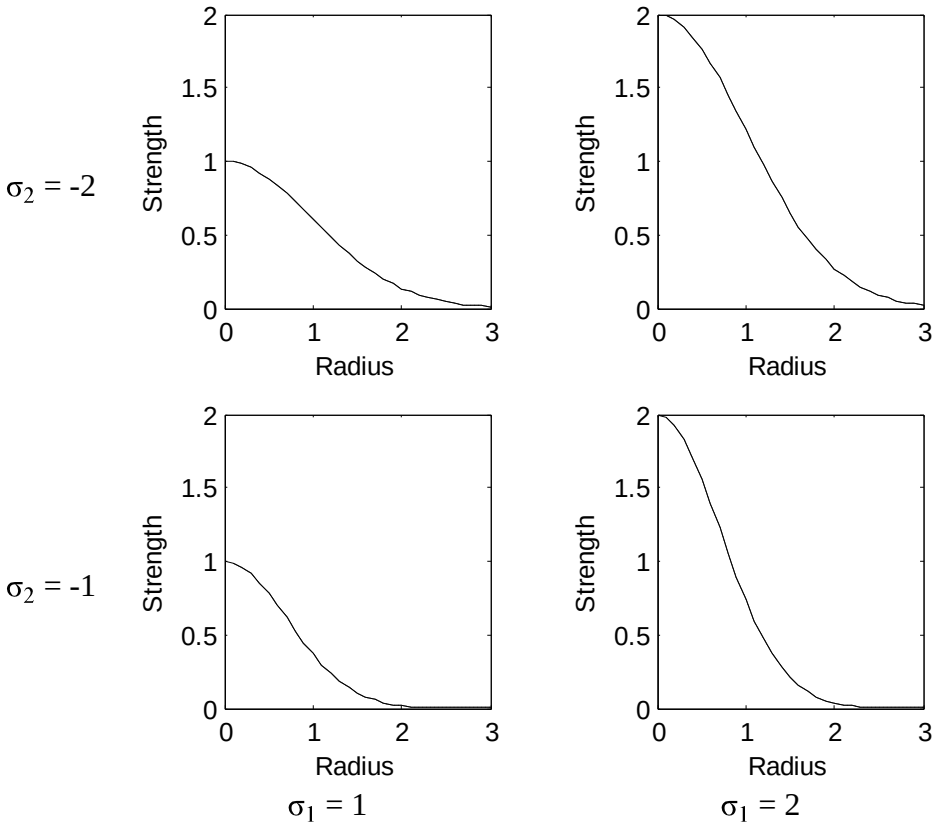


Figure 36 Effects of Changing σ_1 & σ_2 For Repulsive Potential

To get the values of σ_1, σ_2 and σ_4 three points on the potential field are determined and used. The first point is determined using the minimum aircraft turning radius and the magnitude of the gradient required to counteract the goal strength. The minimum radius the aircraft requires to turn was determined by running simulations

where a command was given to turn the aircraft and then recording how long it took the aircraft to turn. From a few runs, 250ft was found to be the turning distance needed for a commanded 90 degree turn. The value of the first coordinate is [250 1] written in [R, $\|\nabla\tau\|$]

For the second point, a buffer distance is added onto the minimum turning radius and half the goal strength is used. The buffer used for this aircraft is 50ft making the second point [300 0.5]. The third point used is set from the minimum altitude above ground the UAV will be allowed to fly. For this aircraft 100ft was chosen from run simulations where commands were given to change the altitude. This makes the third and final point [100 1].

To detemine the constant values, we must consider the strength of the goal potential. The two points are plotted and the constants of the first two points are plotted against the radius in the X and Y direction as shown in Figure 37. The third point is plotted against the Z direction as shown in Figure 38. The values of σ_1, σ_2 and σ_4 were then adjusted by hand until the lines on the two plots lie on the points. The values could also have been numerically solved using an optimization routine like fzero or fminbnd in MATLAB.

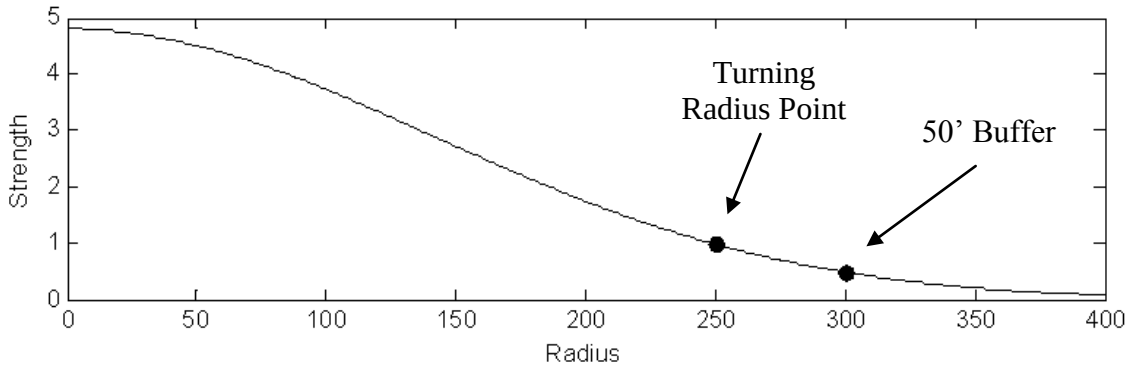


Figure 37 First Two Points Plotted In X & Y Plane As A Function of Radius

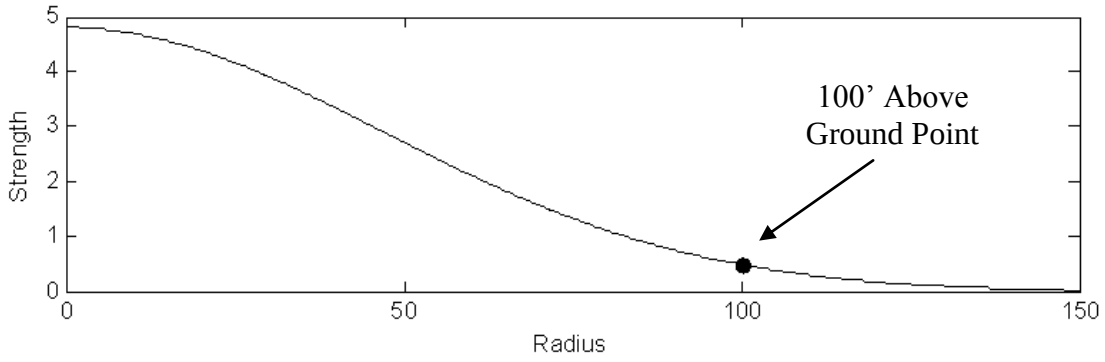


Figure 38 Last Point Plotted In Z Plane As A Function of Radius

This makes σ_1= 4.8, σ_2= -39674.1, σ_4= 3 and gives the following repulsive potential equation.

$$R = \sqrt{X^2 + Y^2 + (3 * Z)^2} \quad \textbf{23}$$

$$\nabla\tau = -4.8 * e^{\left[\frac{1}{-39674}(X^2+Y^2+(3*Z)^2)\right]} * \frac{1}{R} * X\hat{i} - 4.8 * e^{\left[\frac{1}{-39674}(X^2+Y^2+(3*Z)^2)\right]} * \frac{1}{R} * Y\hat{j} - 4.8 * e^{\left[\frac{1}{-39674}(X^2+Y^2+(3*Z)^2)\right]} * \frac{1}{R} * 3 * Z\hat{k} \quad \textbf{24}$$

4.4 Terrain Avoidance Algorithm Conclusion

The potential function algorithm was easy to develop, code and test. One difficulty has been determining how to set the exponential repulsive gradient equation constants. The method developed for setting them in the previous section took a lot of effort to accomplish. Unfortunately, the only true way to know if it will work is to do actual flight testing. One future benefit to the algorithm is that it can later be expanded to handle moving objects that need to be avoided and even areas that are determined to be more dangerous to fly through. Another downside to the algorithm is that it can get trapped in a local minima and be unable to get out, fortunately there are a number of path

methods already developed that can be used to solve this problem including backtracking and random walks [31].

Chapter 5: Results

5.1 Simulation Results

Simulations were performed on the developed terrain avoidance system using the Piccolo Autopilot Command Center and the developed MATLAB code. The main simulation performed was a flight profile that was programmed into the Command Center that had the aircraft fly over San Luis Obispo, CA. The flight path has the UAV flying through the ground in a number of locations to test that the algorithm can successfully redirect the flight path. Figure 39 depicts the path programmed into the Autopilot.

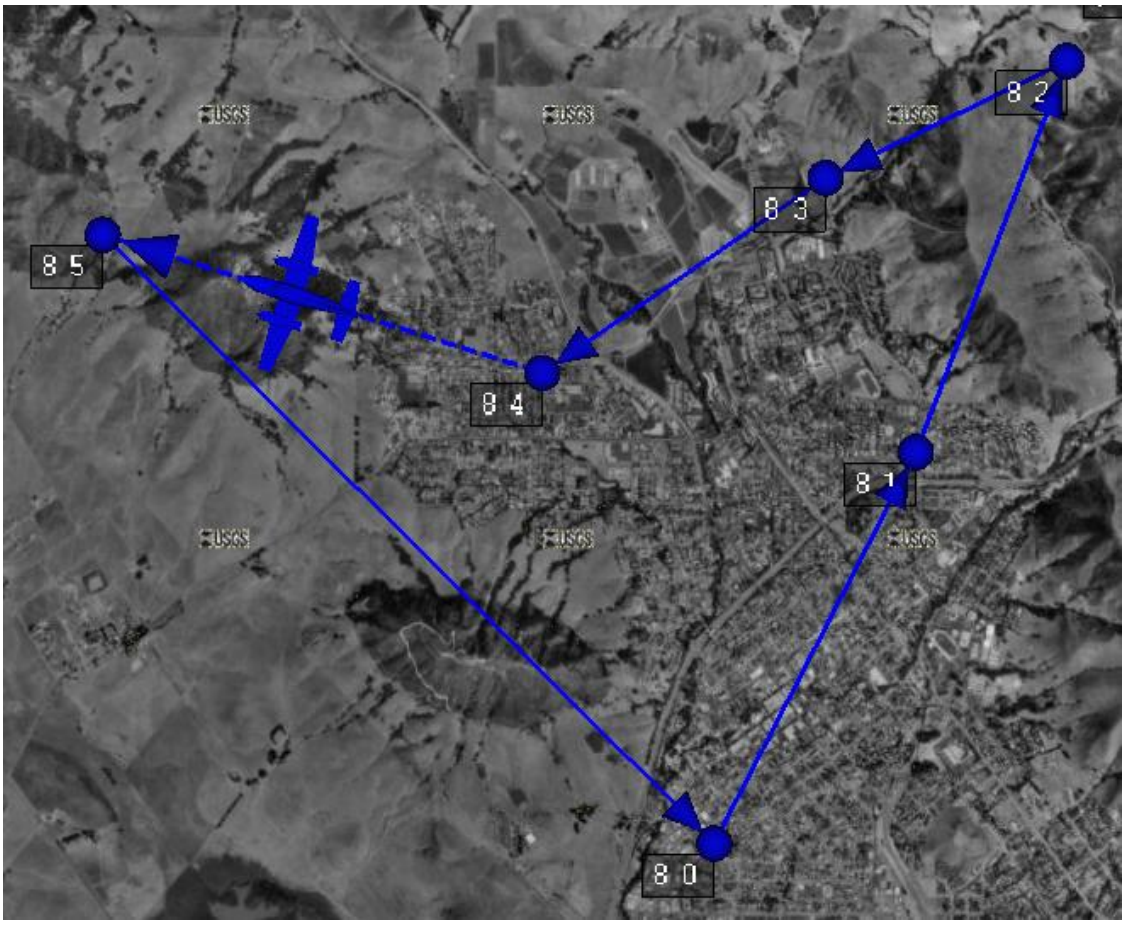


Figure 39 Commanded Flight Path Over San Luis Obispo

The simulation runs showed that the terrain avoidance algorithm was able to fly around all the obstacles most of the time. Unfortunately it would crash every so often due to what we believe is lag in the computer. It is thought that if it were running on the real hardware and not in simulation that it would not lag at all and therefore not fail. One of the successful runs made is shown in Figure 40 where the commanded path is in black and the flown flight path controlled by the algorithm in red.

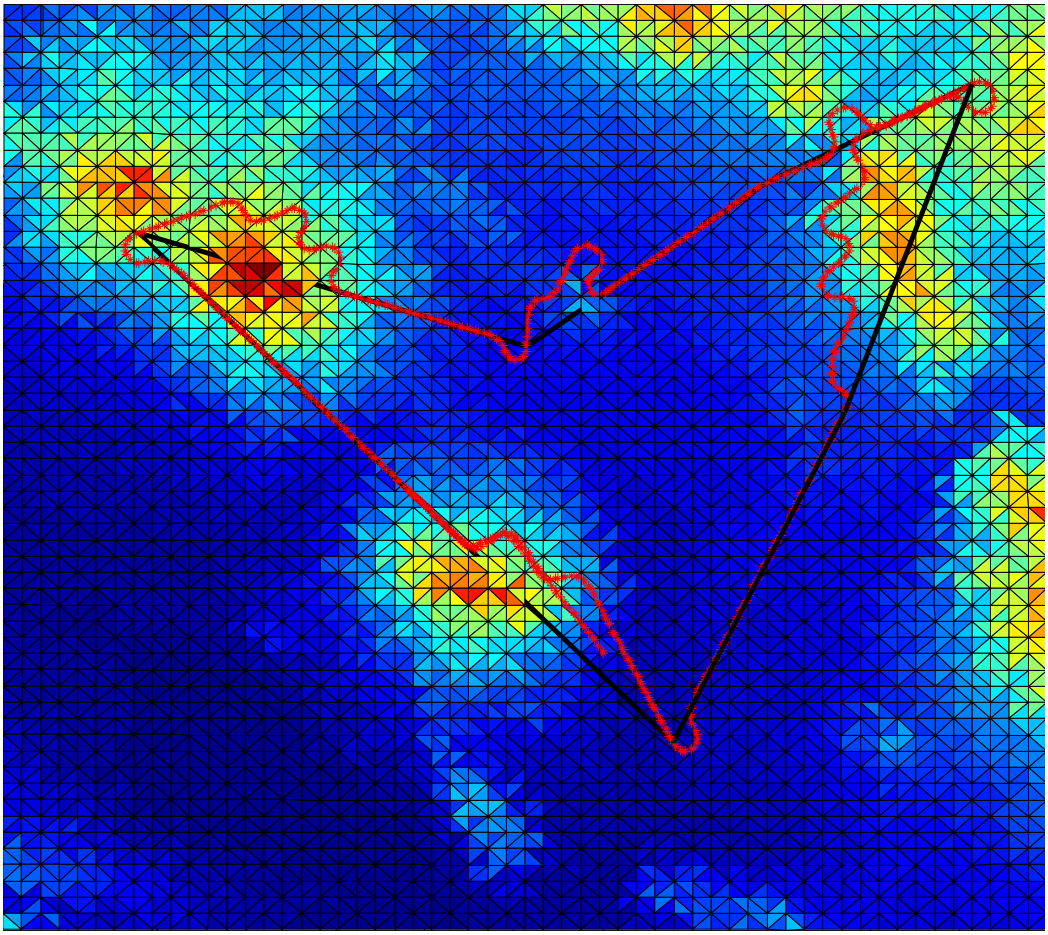


Figure 40 Simulated Flight Path Over San Luis Obispo

5.3 Lesson Learned

The method of giving the autopilot new waypoints as a way to command the vehicle around an obstacle is introducing lag into the system from a controls standpoint. The algorithm knows it needs to deviate, but the autopilot takes too much time to process

the new waypoint. It is believed that if the potential function gradient direction is sent directly into the autopilot's inner loop instead of using waypoints, the terrain avoidance system will be able to react more quickly. This will allow the aircraft to fly very close to terrain, something that currently cannot be done in the simulations.

When tackling a large project like this where an algorithm is being developed, it is better to start with the algorithm and not investigate hardware until the algorithm is well understood. I spent a great deal of time trying to develop a microcontroller and the accompanying libraries only to find out that the microcontroller chosen would never work for this project because I did not understand the requirements of the algorithms.

5.3 Future Work

Now that the algorithm has been developed, the next big step in development is to incorporate hardware in the loop simulations. For this to happen, the microcontroller needs to be selected and the algorithm and matrix method for storing points need to be put onboard. Next, the required communication protocols to talk to the autopilot and laser need to be developed so that the simulations can be performed. Once successful, work can be focused on implementing the system on an aircraft for flight testing.

5.4 CalPoly RMAX Helicopter

Cal Poly received a RMAX Helicopter about a year and a half ago and within the last year an autopilot making it fully autonomous has been installed. Some talk about incorporating a terrain avoidance system onboard so that it can perform search and rescue tasks in the San Luis Obispo, California area. This thesis was developed to be able to incorporate terrain avoidance for an aircraft but the potential method used can also be easily adapted for use by a helicopter if chosen to. Because the helicopter can actually

stop, hover, and climb vertically, the algorithm can actually be significantly more powerful. The gradient and underlying theories can be applied similarly, and provide the most desirable direction of flight. The real benefit is that the strength of the repulsive gradient can be tied to the maximum airspeed of the helicopter so that when it gets close to an object it will slow down. This will allow it to become more maneuverable when it is next to an obstacle as well as travel slower so the sensors are able to get more resolution of the obstacle. Ultimately, this would allow the helicopter to fly very close to terrain which can be very beneficial if it were to be used for search and rescue.

REFERENCES

[1] ”More About Balloons.” Scientific America March 1849

[2] http://www.reference.com/browse/wiki/CFIT. 05 08 2007.

[3] ICAO. "Annex 6, Part 1 CH6 sec 6.15. " November 1995. .

[4] FAA. "AC 25-23. " Airworthiness Criteria for the Installation Approval of a Terrain Awareness and Warning System (TAWS) for Part 25 Airplanes. 2000.

[5] Aeromech Engineer, Inc. Aeromech Engineering Flight Tests New Sharkfin Mission Control System. 3 August 2007. <http://www.aeromechengineering.com/newspress.php?iframesrc=sharkfin>. 2010

[6] Procerus. 8 August 2007. <www.procerusuav.com>.

[7] Micro Pilot. 8 August 2007. <www.micropilot.com>.

[8] O-NAVI. 8 August 2007. <http://o-navi.com>.

[9] Cloud Cap Technology. 3 August 2007 <www.cloudcaptech.com>.

[10] Halper, Ryan. "Integration of an Autopilot System for Autonomous Unmanned Aerial Vehicles. " San Luis Obispo : California Polytechnic State University, 2009.

[11] Drela, Mark et Harold Youngren. "Athena Vortex Lattice. " (2007).

[12] Viquerat, A., L. Blackhall, A. Reid, S. Sukkarieh. "Reactive Collision Avoidance for Unmanned Aerial Vehicles Using Doppler Radar. " 6th International Conference on Field and Service Robotics. Chamonix: France, 2007.

[13] Spark Fun Electronics. June 2010.

[14] Darpa. The DARPA Grant Challenge 2005. 2005. 2010 <http://www.darpa.mil/grandchallenge05/gcorg/index.html>.

[15] OPTI-LOGIC. July 2008 <www.opti-logic.com>.

[16] Servo City. June 2009 <www.servocity.com>.

[17] Santos, Vitor. A 3D Laser Scanner Using A Tilt Unit. 29 October 2009. August 2010 <www2.mec.ua.pt/robotics/scan3d/3dLaserMain-FirstProject.htm>.

[18] Velodyne High Defintion Lidar. June 2010 <http://www.velodyne.com/lidar/products/specifications.aspx>.

[19] Owen, Scott. Ray Tracing. 20 July 1999. May 2009 <http://pratt.siggraph.org/education/materials/HyperGraph/raytrace/rtrace0.htm>.

[20] Seitz, M. Lat/Lon to Elevation. 2009. 2010 <http://www.latlontoelevation.com/dem_consume.aspx>.

[21] Brake, Nick. Interview. San Luis Obispo, 6 May 2009.

[22] Black Fin. June 2009 <www.analog.com>.

[23] Jackins, C.L. et S.L. Tanimoto. "Octtrees and Their Use in Pepresenting Three-Dimensional Objects. " 14, No. 3, p 249-270 (s.d.).

[24] Owens, G. "Developing Street Standards That Allow Flexibility. " TRB Circulat E-C019: Urban Street Symposium. Sacramento, CA, s.d.

[25] Sinopoli, Micheli, M Donato et J. Koo. "Vision based navigation for an unmanned air vehicle. " Proceedings of the IEEE International Conference on Robotics and Automation.pp. 1757-1765 (2001).

[26] Few, E., J. Langelaan et S. Joo. "Adaptive receding horizon control for vision based navigation of small unmanned aircraft. " Proceedings of the 2006 American Control Conference (2006).

[27] Griffiths, S., et al. "Obsticale and terrain, avoidance for miniature aerial vehicles. " (s.d.).

[28] Eliazar, Austin et Ronald Parr. "DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without. " Department of Computer Science Duke University.

[29] Tsuruta, M. "3-D Simulation of Formation Flight Via Potential Function Approach. "

[30] Krogh, B. "A Generalized Potential Field Approach to Obstacle Avoidance Control. " Robotics Research: The Next Five Years and Beyond. Bethlehem, Pennsylvania, 1984.

[31] Latombe, J. C. "Robot Motion Planning. " Boston : Kluwer Academic Publishers, 1991.